ESP8266

I2S Module Description

Version 1.0

Espressif Systems IOT Team

bbs.espressif.com

Table of Contents

1. Feature description

The I2S module of the ESP8266 contains a Tx (transport) unit and a Rx (receive) unit. Both the Tx and the Rx unit have a three-wire interface that includes:

- Clock line;
- Data line;
- Channel selection line (the line for selecting the left or the right channel).

Note:

The clock and data output will stop when 0 is written into the data line.

The transmission direction of the I2S module is shown in Table 1-1.

Table 1-1. transmission direction of the I2S module

	Tx unit	Rx unit
Clock line	output / input	output / input
Data line	output	input
Channel selection line	output	input

Note:

Both the Tx and Rx unit have a separate FIFO, which has a depth of 128 and a width of 32 bits, and can be visited by software directly. You can also make an automatic DMA operation to FIFO by the SLC module.

2. System configuration

2.1. I2S module configuration

2.1.1. Module reset

Bits 0~3 in the ***I2SCONF*** register provide the software reset feature to the I2S. Write 1 and then 0 to complete the reset operation. Different bits are used for:

- Bit 0: I2S_TX_RESET
- Bit 1: I2S_RX_RESET
- Bit 2: I2S_TX_FIFO_RESET
- Bit 3: I2S_RX_FIFO_RESET

2.1.2. Module start

Provide a running clock

To start the I2S module to transport or receive data, firstly you need to provide a running clock for the I2S by invoking the system function below:

```
i2c_writeReg_ Mask_def (i2c_bbpll, i2c_bbpll_en_audio_clock_out, 1)
```

Start the Tx module

Bit 8 in the I2SCONF register is used to start the Tx module.

- In the master Tx mode, when bit 8 is 1，the Tx mode will output the clock signal, the left and right channel signals and data. The first frame data is 0, and then the FIFO data will be shifted out.
  - If no data is written into the the FIFO, the data line will remain 0.
  - If the FIFO has transported all the written data and no new data is written in the FIFO, the data line will loop the last data in the FIFO.
- In the slave passive Tx mode, the Tx module will be started when it receives a clock signal from the Rx module.

Start the Rx module

Bit 9 in the ***I2SCONF*** register is used to start the Rx module. In the master receive mode:

- When bit 9 is 1，the Rx mode will output the clock signal, and sample the data line and the channel selection line.
- When bit 9 is 0, it will stop the clock signal transport.
- In the slave receive mode, it is prepared to receive any data from the master.

2.1.3. Tx/Rx FIFO mode

FIFO access mode

Bit 12 of ***I2S_FIFO_CONF*** defines the access mode of the FIFO.

- When bit 12 is 1, the SLC will make a DMA operation to the FIFO. Direct access to the FIFO will be invalid.
- When bit 12 is 0，the FIFO can be accessed directly by software.
- The default value of bit12 is 1.

Tx FIFO mode

Bits 13~15 of ***I2S_FIFO_CONF*** **are** used to control the transport data format for ***i2s_tx_fifo_mod*****.**

Value	Description
0	16bits_per_channel full data (dual channel, FIFO data organisation, 16 bits data in the left channel,16 bits data in the right channel, and 16 bits data in the left channel)
1	16bits_per_channel half data (single channel, FIFO data organisation, 16 bits data, 16 bits invalid , 16 bits data)
2	24bits_per_channel full data discontinue (dual channel, FIFO data organisation, 24 bits data in the left channel, 8 bits invalid, 24 bits data in the right channel, 8 bits empty)
3	24bits_per_channel half data discontinue (single channel, FIFO data organisation, 24 bits data, 8 bits invalid, 24 bits data, 8 bits empty)
4	24bits_per_channel full data continue (left and right channels, FIFO data organisation, 24 bits data in the left channel, 24 bits data in the right channel)
5	24bits_per_channel half data continue (single channel, FIFO data organisation, 24 bits data, 24 bits data)
6~7	Invalid

RX FIFO mode

Bits 16~18 of ***I2S_FIFO_CONF*** is used to control the receive data format for ***i2s_rx_fifo_mod***.

Value	Description
0	16bits_per_channel full data
1	16bits_per_channel half data
2	24bits_per_channel full data discontinue
3	24bits_per_channel half data discontinue
4~7	Invalid

2.1.4. Channel mode

Tx channel mode

Bits 0~2 in the ***I2SCONF_CHAN*** are used for the Tx channel mode (tx_chan_mod).

Value	Description
0	Dual-channel
1	Right channel (left and right audio channels are used to put the data of the right channel)
2	Left channel (left and right audio channels are used to put the data of the left channel)
3	Right channel (put a constant from regfile in the left channel)
4	Left channel (put a constant from regfile in the right channel)

Rx channel mode

Bits 3~4 in the ***I2SCONF_CHAN*** are used for the Rx channel mode (rx_chan_mod).

Value	Description
0	Dual-channel
1	Right channel
2	Left channel

2.1.5. Clock mode

in the ***I2SCONF*** :

- Bits16~21 are the prescaler of the input clock (I2S_CLKM_DIV_NUM).
- Bits22~27 are the frequency divider of the communication clock signal (I2S_BCK_DIV_NUM).

2.1.6. Other configurations

Register I2SRXEOF_NUM sets the number of data to be received when the Rx FIFO triggers the SLC transport (unit: 4 bytes).

See the definitions of ***i2s_reg.h*** in DEMO. Other instructions will be updated.

2.2. Link list configuration

In the ESP8266, the DMA transfers the receive and transport packets in the SDIO to the corresponding memory. The software will define the structure (or group) of the registration list and cache space(s).

As shown in Figure 2-1, there is only one cache space and one registration list. Write the first address of the cache and other information to the registration list, and then write the first address of the registration list to the hardware register of the ESP8266. Therefore, the DMA will automatically operate the SDIO and the cache space.

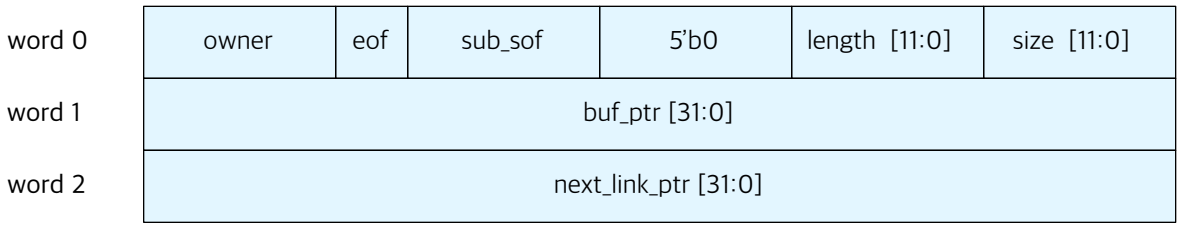

Figure 2-1. Registration list

Field name		Description
owner	1'b0	Software operates the buffer of the current link. The MAC shouldn't use this bit.
	1'b1	Hardware operates the buffer of the current link.
eof		Flag of frame end（for the end of AMPDU sub-frame，the mark isn't needed). ▸ When the MAC transports the frames, it's used in the end of the frame. For the link in the position of eof，the buffer_length[11:0] should be equal to the length of the remaining frame; otherwise, the mac will report an error. ▸ When the MAC receives the frames, it's used to indicate that the frame has been received completely and the value is set by hardware.
sub_sof		Flag of sub-frame start. It's used to differentiate different sub-frames in the AMPDU. It's only for MAC transport.
length[11:0]		The actual size of the buffer.
size[11:0]		The total size of the buffer.
buf_ptr[31:0]		The start address of the buffer.
next_link_ptr[31:0]		The start address of the next descripter. When the MAC is receiving the flame, the value is "0", indicating that there is no empty buffer to receive any flames.

2.3. SLC module configuration

2.3.1. Basic configuration

The SLC module provides the ESP8266 with DMA service of several modules.

Follow the instructions below so that the SLC module is used for the FIFO transmission of I2S.

- Set Bits 12~13 (SLC_MODE) of the ***SLC_CONF0*** to 01.
- Set Bit 17 (SLC_INFOR_NO_REPLACE) and Bit 16 (SLC_TOKEN_NO_REPLACE) of the ***SLC_RX_DSCR_CONF*** to 01.

2.3.2. Write the first address

Bits 0~19 of ***SLC_RX_LINK (SLC_TX_LINK)*** register are the first 20 bits of the Rx (Tx) registration list address. The first address of the registration list should be written to be the register before the SLC hardware is statred.

2.3.3. Start the SLC transmission

Bit 29 of ***SLC_RX_LINK (SLC_TX_LINK)*** register is the control bit for starting the SLC transmission. In the cache space, register a link list and write the first 20 bits of the link table address to the hardware, and then set bit 29 to 1 to start the SLC transmission.

3. API function description

The following functions can be found in:

```
/app/driver/i2s.c and /app/include/driver/i2s.h
```

3.1. Void function

3.1.1. void i2s_test

Function	void i2s_test(void)
Feature	I2S Programs for read and write testing of the module. It is the core function of the DEMO, which can be used to test the transporting and receiving communications of the I2S.
Parameter	null

3.1.2. void i2s_init

Function	void i2s_init(uint8 slc_en)
Feature	Configure the related registers of the I2S.
Parameter	slc_en: Enable the SLC module access. When it's 0, the software will operate the FIFO, For other values for the SLC module directly access FIFO, refer to 2.1.3. ***Tx/Rx FIFO mode***.

3.1.3. void creat_one_link

Function	void creat_one_link (uint8 own, uint8 eof,uint8 sub_sof, uint16 size, uint16 length, uint32* buf_ptr, uint32* nxt_ptr, struct sdio_queue* i2s_queue)
Feature	Set up a link register structure.
Parameter	struct sdio_queue* i2s_queue: The first address to be configured structure space. For details of other parameters, refer to ***2.2. Link list configuration***.

3.1.4. void slc_init

Function	void slc_init (uint8 trans_dev)
Feature	Basic configuration of the SLC module. For configuration instructions, refer to ***2.3. SLC module configuration.***
Parameter	uint8 trans_dev: SLCModule access device, 1 is I2S， 0 is SDIO， other input values are not valid.

3.2. CONF function

3.2.1. CONF_RXLINK_ADDR

Function	CONF_RXLINK_ADDR(addr)
Feature	Configure the Rx link list address to the register. For configuration instructions, refer to ***2.3. SLC module configuration.***
Parameter	addr: link list address

3.2.2. CONF_TXLINK_ADDR

Function	CONF_TXLINK_ADDR(addr)
Feature	Configure the TX link list address to the register. For configuration instructions, refer to ***2.3. SLC module configuration.***
Parameter	addr: link list address

3.3. START function

3.3.1. START_RXLINK

Function	START_RXLINK()
Feature	Start the Rx transmission of the SLC module. For configuration instructions, refer to ***2.3. SLC module configuration.***
Parameter	null

3.3.2. START_TXLINK

Function	START_TXLINK()
Feature	Start the Tx transmission of the SLC module. For configuration instructions, refer to ***2.3. SLC module configuration.***
Parameter	null